arXiv:2512.24522v1 [math.PR] 30 Dec 2025

使用与图的最大度数成线性关系的颜色数，在图上以线性时间进行合法着色

KritikaBhandari Mark Huber

2026年1月1日

摘要

本文提出了一种从图的合法着色集合中进行精确采样的新算法。这是首个能够保证期望运行时间与最大度数为 Δ 的图的规模呈线性关系的此类算法，条件是颜色数大于 $3.637\Delta + 1$。

1 引言

图的合法着色是指从一个集合 $C = \{1, \ldots, k\}$ 中为每个节点指定一个颜色，使得图中每条边的两个端点被指定为不同的颜色。也就是说，对于图 $G = (V, E)$ 和颜色集 C，$x : V \to C$ 是合法着色，意味着 $(\forall \{i, j\} \in E)(x(i) \neq x(j))$。如果图 G存在合法着色，就称 G 是 k可着色的。

Guthrie [4]在研究平面图的四色可着色问题时引入了这种着色。Karp [11]证明，对于 $k \geq 3$，判定一般图是否可 k 着色是一个 NP完全问题。Jerrum [9] 证明（同样在 $k \geq 3$ 的情况下），计算图的合法着色数量是 #P完全问题。

令 Δ 为图的最大度数。当 $k \geq \Delta + 1$ 时，图必定存在一种着色，因此存在性问题不再是 NP 完全的。然而，相关的计数问题仍然是 #P 完全的 [9]。因此，接下来考虑如何在给定相对误差 $\epsilon > 0$ 范围内近似计算正常着色的数量。

Jerrum、Valiant 和 Vazirani 提出了一种方法：在能够从原问题及其所有约简问题中近似均匀地抽样的前提下，为计数自可约组合对象设计随机近似算法 [10] 。幸运的是，正常着色属于这类自可约问题。在 [9]„ Jerrum 证明，当 $k > 2\Delta$ 时，平稳分布在正常着色集合上均匀分布的简单吉布斯马尔可夫链（也称为 Glauber 动力学）通过耦合论证可实现快速混合。因此，可以使用该马尔可夫链，从具有 n 个节点的图的 k-着色中进行近似均匀抽样，所需步数为 $\Theta(n\ln(n))$，其中每一步耗时 $O(\Delta)$。

Vigoda [15] 随后提出了一种新的马尔可夫链，当 $k > (11/6)\Delta$ 时，该链需要经过 $\Theta(n\ln(n))$ 步才能混合；然而，每一步都需要改变 $O(n)$ 个节点的颜色，因此总运行时间为 $O(n^2\ln(n))$。Chen 等人 [2], 使用 Vigoda 提出的同一条链，将这一 $11/6$常数降低到了 $(11/6) - \epsilon_0$，其中 $\epsilon_0 > 0$。

在开发近似采样方法的同时，精确采样技术也取得了进展。1996 年，Propp 和 Wilson 提出了一种名为从过去耦合（CFTP） [13] 的完美采样协议。该协议提供了一种通用方法，用于构造能够从马尔可夫链的平稳分布中精确生成样本的算法；这些算法在链中运行的步数是随机的。该方法可自动应用于单调马尔可夫链；遗憾的是，Jerrum 和 Vigoda 使用的链并不属于这一类。

1998年，Huber [6] 提出了界链的概念，它与原始马尔可夫链并行运行，使 CFTP 得以应用于该链。该方法被应用于 Jerrum 使用的着色链，从而在 $k > \Delta(\Delta + 2)$时，用时$\Theta(n\ln(n))$，便可从合法着色集合中获得精确均匀的样本。

这一关于颜色数量的上界与图的最大度数呈二次关系，Bhandari 和 Chakraborty [1]终于在2020年打破了它。他们只需 $k > 3\Delta$ ，便能达到 $\Theta(n\ln(n))$的期望运行时间。他们再次使用了界链方法。随后，Jain、Sah 和 Sawhney [7]进一步改进了界链方法，只需$k > ((8/3)+o(1))\Delta$ 种颜色，即可获得平均运行时间为 $\Theta(n\ln(n))$的算法。

现在考虑用于生成正常着色的线性时间算法。目前尚无使用线性时间的近似采样方法。一个随机-

性回收器风格的算法已在 [3], 中给出，但要保证期望运行时间为线性，该算法需要 $\Omega(\Delta^4)$ 种颜色。

随机性回收器（RR）算法是对冯·诺伊曼在 1951 [12] 正式提出的接受-拒绝法，以及 Propp 和 Wilson 从 1998 [14]提出的弹出技术的扩展。其思路是先从一个比目标分布更容易处理的分布中生成样本，然后以特定概率接受该样本作为目标分布的一个实现值，从而使接受后样本的分布与目标分布相同。在基本的接受-拒绝法中，如果样本未被接受，就会被彻底丢弃。

在 RR 方法中，我们不会彻底丢弃样本，而是尽可能多地保留（回收）样本。确实，拒绝操作使样本偏向了一个新的分布，但这并不意味着样本中剩余的随机性不多。

因此，RR 算法不仅跟踪状态，还跟踪描述该状态分布的未归一化权重集。算法从一个易于采样的权重集开始，然后在每一步以一致的方式改变状态、权重，或同时改变两者。

RR算法的运行过程如下。

1. 从初始权重集中抽取一个状态。

2. 不断更新状态或权重，直到这些权重是目标权重的倍数。第2节将探讨有效的更新方式。

3. 输出最终状态。

如 [3, 5], 所示，最终状态的分布将与目标分布完全相同。

本文的主要成果是实现了 RR 协议，由此得到了首个能在线性时间内对图进行正常着色采样的算法，其中颜色数量与最大度数成线性关系。

定理 1。 存在一种随机性回收算法，可生成一个样本，该样本是从具有 n 个节点且最大度数为 Δ 的图的合法着色中精确均匀抽取的；该算法平均使用 $\Theta(n)$ 个步骤，每步执行时间为 $O(\Delta)$，并且最多使用 $O(\Delta \log(k))$ 个均匀随机比特，其中 $k > 3.637\Delta + 1$。

与所有完美模拟算法（不同于近似采样算法）一样，即使不满足关于 k 的这一界限，也可以运行该算法。如果算法终止，所得结果就是一个精确样本。该条件仅用于在运行算法之前证明其期望运行时间为线性。

本文其余部分安排如下。下一节回顾 RR 类算法的基本概念，并介绍用于合法着色的新算法。最后一节将分析该算法并给出主定理。

2 算法

由于该算法使用 RR 协议，我们将首先讨论这种方法的一般形式。

2.1 RandomnessRecycler 协议

术语权重用于指代任意未归一化密度。例如，考虑权重集 $w(x)$，当 $x \in \Omega$时其值为 1，否则为 0。这可以用示性函数 $w(x) = \mathbb{I}(x \in \Omega)$表示。该权重集编码了 Ω 上的均匀分布。

若 $w_1 = \alpha w_2$ 对某个正常数 α成立，则称 w_1 与 w_2 等价。由于权重未经归一化，两个等价的权重函数表示相同的分布。

RR算法通过执行两种能够改变权重的步骤之一来运行。

1）使用转移函数 $p(x, y)$ 随机改变状态。此操作根据马尔可夫链步骤的标准公式改变该状态的权重：

$$w(x) \mapsto w'(y) = \sum_{x'} w(x')p(x', y).$$

2）随机改变权重。如果当前权重函数为 $w(x)$，则以概率 $p_{\text{acc}}(x)$ 转移到权重 $w(x)p_{\text{acc}}(x)$，并以概率 $1 - p_{\text{acc}}(x)$ 转移到权重 $w(x)(1 - p_{\text{acc}}(x))$。这可以看作将权重函数 $w(x)$拆分为两个权重函数。

如果 $p_1(x) \dots, p_\ell(x)$ 是非负函数，并且对于每个x，它们的和均为 1，那么以概率 $p_i(x)$ 转移到权重 $w(x)p_i(x)$。这可以看作将权重函数拆分为 ℓ 种不同的可能性。

使用RR算法需要两种特殊权重。令 w_{init} 为一个易于采样的权重函数，令 w_{tar} 为目标权重函数。

算法从权重 w_{init}中抽取一个状态作为起始状态。每一步都会使用随机改变或分裂步骤。通常会选择 $p(x)$，使 $w(x)p(x)$更接近目标分布。这意味着 $w(x)(1 - p(x))$通常会离该分布更远。

当权重函数与目标权重函数一致时，算法终止。最终状态将是从最终权重所给出的分布中精确抽取的样本。有关这一结果的证明，请参见 [5] 的第8章。（该章对一般状态空间证明了这一结果，而对于有限空间，该结果可简化为更简单的形式。）

2.2 合法着色的RR算法

要构建用于合法着色的RR算法，需要一组足够广泛、能够处理目标分布的索引，一个易于从中采样的初始分布，以及对接受和拒绝过程中所用权重函数集合进行索引的能力。

在权重函数中，每个节点可能被禁用某种颜色、冻结为某种颜色、可以与相邻节点使用相同颜色，或必须使用与相邻节点不同的颜色。

这些属性将通过状态 x^* 进行编码，该状态为每个节点分配一个值，取自

$$A = \{-k, -k+1, \dots, -1, 0, 1, 2, \dots, k\} \cup \{\perp\}.$$

因此 $x^* \in \Omega^* = A^V$。

1. **禁用颜色** 如果不允许为节点 v 分配颜色 b，则称节点 v 被禁用 使用颜色 b。

   在索引中，将 $x^*(v) = -b$设置为 $b \in C$，以表示节点 v 被禁用使用颜色 b。

2. **固定为某种颜色** 如果节点 v 必须被赋予颜色 c，则称节点v 被冻结为颜色 c。

在索引中，将 $x^*(v) = c$ 设置为 $c \in C$，表示节点 v 被固定为某种颜色 c。

3. **忽略** 如果计算着色是否恰当时，权重不考虑节点 v，则称该状态忽略 节点 v。

记作 $x^*(v) = 0$，表示对于 v 的任意邻居，$x(v)$ 都可以等于 $x(w)$。

4. **不受限制** 如果一个节点不符合前述三个条件中的任何一个，则该节点不受限制。这样的节点不能与邻居具有相同的颜色。

如果节点的行为与普通节点相同，则记作 $x^*(v) = \perp$。此时，$x(v)$ 可以是 C 中的任意元素，并且对于 v 的所有邻居 w，$x(v) \neq x(w)$。

回忆一下，对于示性函数 $\mathbb{I}(p \wedge q) = \mathbb{I}(p)\mathbb{I}(q)$，因此可以通过将各自的示性函数相乘来满足多个要求。

禁用颜色节点的权重因子很简单，只需在 $x^*(v) = -b$ 时设定 $\mathbb{I}(x(v) \neq b)$。固定颜色节点使用 $\mathbb{I}(x(v) = x^*(v))$。忽略节点完全不影响权重。

最后，只要一条边的端点中至多有一个被冻结，且两个端点都未被忽略，就会强制要求这两个端点的颜色不同。对于边 $e = \{v, w\}$，这可以写成

$$f_e(x, x^*) = \mathbb{I}((x(v) \neq x(w)) \vee (x^*(v) = x(v) = x^*(w) = x(w)) \vee (x^*(v) = 0 \vee x^*(w) = 0)).$$

要成为某个索引的有效着色，正确的节点必须冻结为指定颜色，正确的节点必须禁用某些颜色；对于由一条边连接的两个节点，要么其中一个或两个节点被忽略，要么它们必须被赋予不同的颜色。

设 $-C = \{-1, -2, \ldots, -k\}$。与索引 x^* 和状态 $x \in C^V$ 相关联的权重为

$$w_{x^*}(x) = \left[\prod_{v:x^*(v)\in C} \mathbb{I}(x(v) = x^*(v))\right] \left[\prod_{v:x^*(v)\in -C} \mathbb{I}\left(x(v) \neq -x^*(v)\right)\right] \left[\prod_{e\in E} f_e(x, x^*)\right]$$

由于 $w_{x^*}(x) \in \{0,1\}$（并且至少存在一种权重为正的着色），因此这是均匀分布的权重函数。将有三个函数以状态 x 开始，该状态从权重 $w_{x^*}(x)$ 中抽取，并返回一个状态 y，该状态从权重 $w_{y^*}(y)$ 中抽取。在这些函数的执行过程中，权重可能会暂时偏离均匀分布，但函数结束时，权重会回到某个索引权重。

由于每个索引权重函数都是若干指示函数的乘积，因此总权重函数在该索引下所有权重为正的状态上是均匀的。

一个有用的记号是

$$a(x^*) = \{x' \in C^V : w_{x^*}(x') = 1\}$$

这样，给定 x^* 时，x 在 $a(x^*)$ 上服从均匀分布。

初始索引对所有 v 都有 $x_0^*(v) = 0$。注意

$$w_{x_0^*}(x) = \mathbb{I}(x \in C^V).$$

也就是说，它在所有着色上均匀分布，而不仅仅是在正常着色上。这个分布很容易采样：只需从 C 中均匀且独立地抽取每个 $x(v)$。

如果存在一个节点 v 被禁用某种颜色，首先尝试移除这一限制。接下来，如果有一个节点 v 固定为某种颜色，尝试移除这一限制。最后，如果有一个节点 v 被忽略，尝试移除这一限制。

继续直到到达最终索引 x^*_{tar}，此时每个节点的值均为$\perp$。这对应于权重

$$w_{x^*_{\mathrm{tar}}}(x) = \prod_{\{i,j\} \in E} \mathbb{I}(x(i) \neq x(j)),$$

这是正常着色均匀分布的权重函数。

该算法可写如下。

合法着色的 RR 步骤(x, x^*)

1. 如果存在 $v \in V$，使得 $x^*(v) = -b$，其中 $b \in C$，则输出**移除禁用**(x, x^*, v) 并退出。

2. 否则，如果存在 $v \in V$，使得 $x^*(v) = c$，其中 $c \in C$，则输出**移除冻结**(x, x^*, v) 并退出。

3. 否则，如果存在 $v \in V$ 使得 $x^*(v) = 0$，则输出 **移除忽略项**(x, x^*, v) 并退出。

4. 否则，输出 (x, x^*) 并退出。

给定这个单步算法，主算法的运行如下。

用于合法着色的 RR 算法(x, x^*)

1. 对所有 $v \in V$，令 $x^*(v) = 0$。

2. 对每个 $v \in V$，从 C 中均匀且独立地抽取 $x(v)$。

3. 当 $x^* \neq (\perp, \ldots, \perp)$ 时，用 **合法着色的 RR 步骤**(x, x^*) 的输出替换 (x, x^*)。

4. 输出 (x, x^*)。

因此，算法的核心在于如何从索引中移除禁用、冻结和忽略条件。这些步骤必须维持一个不变量才能奏效。

Let

$$(y_1, y_1^*) = \textsf{Remove forbidden}(y, y^*, v)$$
$$(y_2, y_2^*) = \textsf{Remove frozen}(y, y^*, v)$$
$$(y_3, y_3^*) = \textsf{Remove ignored}(y, y^*, v)$$

需要维持的不变量是：若有一个节点 v with$x^*(v) \in -C$，那么所有满足 $x^*(w) \in -C$ 的节点 w 都必须具有 $x^*(v) = x^*(w)$。换句话说，在任意时刻，至多只有一种颜色会被禁止赋给某些节点。

备注：在每一步中如何选择节点 v，对于算法的运行并不重要。选择方式可以是随机的，也可以采用确定性方法，例如给节点编号并始终选择标签最小的节点。任何此类方法都能得到正确的算法，其期望运行时间如定理 1 所述。

从最后一个过程开始，移除一个忽略节点。

2.2.1 移除忽略项

假设 x^* 只有忽略节点或不受限制的节点，且节点 v 被忽略。也就是说， $x^*(v) = 0$ ，并且对于所有 $w \neq v$， $x^*(w) \in \{0, \perp\}$。

令 r^* 与 x^* 为相同的索引，但移除 v处的忽略限制。也就是说， $r^*(V \setminus \{v\}) = x^*(V \setminus \{v\})$，且 $r^*(v) = \perp$。注意，对于任意状态 x。

$$w_{r^*}(x) = w_{x^*}(x) \prod_{w:\{v,w\}\in E} \mathbb{I}(x(v) \neq x(w)) \leq w_{x^*}(x).$$

因此，将以 x^* 为索引的权重转换为以 r^* 为索引的权重，最简单的方法是以概率 $\prod_{w:\{v,w\}\in E} \mathbb{I}(x(v) \neq x(w))$ 接受结果。这种表述等价于：当且仅当 v 没有颜色与 v相同的邻居时，才接受。

如果接受，一切顺利。但如果拒绝，新的权重为

$$w_{x^*}(x)\left[1 - \prod_{w:\{v,w\}\in E} \mathbb{I}(x(v) \neq x(w))\right] = w_{x^*}(x)\mathbb{I}\left(\left[\sum_{w:\{v,w\}\in E} \mathbb{I}(x(v) = x(w))\right] > 0\right)$$

因此，新的权重在节点 v 确实有一个颜色相同的邻居的着色中均匀分布。

遗憾的是，这个条件无法用索引表示。但可以做的是：对节点 v 的未忽略邻居进行均匀随机的不放回搜索，直到找到一个满足 $x(v) = x(w)$的邻居 w 。

在找到这个 w 之前搜索过的邻居不能使用颜色 x（v），而 w 必须使用颜色 x（v）。这可以通过禁止搜索过的邻居使用颜色 x（v），并将节点 w 冻结为颜色 x（v），在索引中表示出来。

此时，将节点 v 的索引设为 0。最后，从 C 中均匀地重新抽取 v 的颜色。

所以现在唯一要做的就是找到一个索引 s^* ，使得

$$w_{s^*}(x') = \mathbb{I}(x' \in a(x^*) \setminus a(r^*)).$$

请思考这个指示函数表达了什么。当着色满足 x^* 的条件、却不满足 r^* 的条件时，它的值为 1。只有当某个未忽略邻居 w 满足 $x(v) = x(w)$ 时，才会出现这种情况。

要找到这个邻居，就以均匀随机、不放回的方式搜索 v 的邻居，直到找到与颜色 $x(v)$ 匹配的邻居 w 。在找到 w 之前搜索过的每个邻居都不具有该颜色：可以通过禁止该节点使用这种颜色来记录这一点。最后，必须将 w 冻结为该颜色。此时，可以将 w 冻结为该颜色。

结合这些想法，得到以下算法。

移除忽略项（x， x^*， v）

1. Let $r^*(V \setminus \{v\}) = x^*(V \setminus \{v\})$ and $r^*(v) = \perp$.

2. 对 v 的非忽略且未冻结邻居进行均匀随机的不放回搜索。

3. 如果 v 没有非忽略且未冻结的邻居具有颜色 x（v），则输出（x， r^*）并退出。

4. 否则，执行以下操作：

   a. 令 $W = \{w_1, \ldots, w_\ell\}$ 为在找到具有 x（w）$= x$（v）的非忽略邻居 w 之前，均匀随机不放回搜索过的邻居集合。b. 令 s^*（$V \setminus$（$W \cup \{v, w\}$））$= x^*$（$V \setminus$（$W \cup \{v, w\}$））， s^*（W）$=$ $\{-x$（v）$\ldots$， $-x$（v）$\}$， s^*（w）$= x$（w），s^*（v）$= 0$。c. 令 s（$V \setminus \{v\}$）$= x$（$V \setminus \{v\}$）。从 C 中均匀抽取 s（v）。d. 输出（s， s^*）并退出。

引理 1. 移除忽略项 对输出状态的权重函数由输出索引给出。

证明。 设 (y, y^*) 是 **移除忽略项**(x, x^*) 的输出。

情况 1：$(y, y^*) = (x, r^*)$。由于此步骤中状态没有改变，因此输出权重就是

$$w_{x^*}(x)\left[\prod_{w:\{v,w\}\in E} \mathbb{I}(x(v) \neq x(w))\right] = w_{r^*}(x)$$

并且该情况已经完成。

情况 2：假设 $(y, y^*) = (s, s^*)$。若 $s \notin a(s^*)$，则要么 $w_1, \ldots, w_\ell$ 中有一个具有颜色 $x(v)$，并且在搜索中会被指定为 w 的角色；要么 w 不具有颜色 $x(v)$，因此不会被选为具有颜色 $x(v)$ 的邻居。在这两种情况下，s 都不可能是输出，因此 $w_{s^*}(s) = 0$，如所需。

现在考虑 $s \in a(s^*)$ 的情况。那么从状态 x 移动到状态 s 的概率是多少？虽然 s^* 告诉我们在找到颜色冲突的邻居之前搜索过的状态集合 $\{w_1, \ldots, w_\ell\}$，以及 w，但它没有揭示搜索这些状态的顺序。因此，进行这次特定搜索，随后选择特定的 $s(v)$ 的概率为

$$\frac{\ell!}{\deg(v)(\deg(v)-1)\cdot(\deg(v)-\ell+1)}\frac{1}{\deg(v)-\ell}\cdot\frac{1}{k}.$$

因此

$$w_{s^*}(s) = w_{x^*}(s)\frac{\ell!}{\deg(v)(\deg(v)-1)\cdot(\deg(v)-\ell+1)}\frac{1}{\deg(v)-\ell}\cdot\frac{1}{k}.$$

特别地，当 $s \in a(s^*)$ 时，$w_{s^*}(s)$ 是一个固定的正常数。因此，可以对这组权重进行归一化。

由于在存在 v 的邻居 w，使得 $x(v) = x(w)$ 的 s 上，$w_{x^*}(s)$ 为常数，且其他因子不依赖于 s，因此对于每个权重为正的状态 s，权重 $w_{s^*}(s)$ 都相同。因此，输出状态 y 在 $a(y^*)$ 上均匀分布。

□

2.2.2 移除冻结节点

这是三个步骤中最简单的一个，因为如果没有禁用节点，那么可以将冻结节点 v 转换为忽略节点，并将 v 的所有未冻结且未忽略的邻居设为禁用，使其不能采用颜色 $x(v)$。随后，可以从 C 中均匀地重新抽取 v 的颜色，并将其状态重新索引为忽略。

对于任意节点 v，令 $N(v)$ 表示 v 的邻居集合。

移除冻结(x, x^*, v)

1. 令 r^* 为索引，在该索引处节点 v 被更改为忽略状态，而 v 的邻居中未冻结且未被忽略的邻居则被设为禁用颜色 x（v）。

2. 令 r 为一种状态，除 v 外的所有位置都等于 x ，并从 C 中均匀随机地选取 r（v）。

3. 输出（r， r^*）。

移除冻结 的权重函数由输出状态的输出索引给出。

根据 x^*，已知 v 的冻结颜色。因此，对于给定的状态 r，恰好存在一个状态 x，它以概率 $1/k$转移到 r。这意味着

$$w_{r^*}(r) = w_{x^*}(x)\frac{1}{k} = \mathbb{I}(r \in a(r^*))\frac{1}{k}.$$

因此， w_{r^*}（r）等价于 a（r^*）上的均匀权重，输出是正确的。

2.2.3 移除禁用

最难处理的限制是节点 v 被禁止使用颜色 $b = -x^*$（v）的情况。在这种情况下，必须尝试将 x（v）的颜色切换为 b。这样做的概率需要足够高，以便后续的接受/拒绝步骤能够使所有权重都变得相同。

开始之前，如果 v 的某个邻居已冻结为颜色 b，那么可以直接移除 b 的禁用颜色条件，且完全不改变允许状态集合。这是简单的情况。

对于较难的情况，首先定义以下集合和常数。

$n_1(x,\ x^*) = v$ 在索引 x^* 下处于状态 x 时的有效颜色数量 $n_2(x^*) = v$ 的冻结邻居未使用的颜色数量 $n_3(x^*) = v$ 中既未冻结也未忽略的邻居数量 $d(x^*) = n_2(x^*) - n_3(x^*)U(x^*) = v$ 中既未冻结也未忽略的邻居集合

注意， $d(x^*) - 1$ 表示可供 v使用的颜色数量下界。也就是说，

$$n_1(x, x^*) \geq d(x^*) - 1.$$

生成一个新状态 r，除了节点 v处（可能）不同外，其他各处都与状态 x相同。以概率 p_1，设置 $r(v) = b$；以概率 $1 - p_1$，则令 $r(v)$保持为 $= x(v)$。注意，$n_1(x, x^*) = n_1(r, x^*)$，因为它不依赖于节点 v的颜色。

令除 v 外的每个节点上的 r^* 都与 x^* 相同，并设 $r^*(v) = \perp$。那么，对于 $a(r^*)$ 中的每个状态，r 的权重都为正；但它也会给满足 $x^*(V \setminus \{v\})$ 上所有限制、却允许 v 与一个邻居同时取颜色 b 的状态赋予正权重。将这组状态称为 a_1。那么$a(r^*) \subseteq a_1$，所以 $\mathbb{I}(r \in a(r^*)) \leq \mathbb{I}(r \in a_1)$。

此外，这些权重并不均匀。对于给定的着色 r，如果$r(v) \neq b$，那么只有一个状态 x' 可以转移到 r。但请注意，对于 $r(v) = b$，有 $n_1(r, x^*)$ 个不同的状态 $x' \in a(x^*)$ 可以转移到 r。

因此，这一步之后的权重将为

$$\begin{aligned} w_1(r) &= \mathbb{I}(r \in a_1) \left[(1-p_1)\mathbb{I}(x(v) = r(v)) + p_1 \sum_{x' \in a(x^*): x'(V\setminus\{v\}) = r(V\setminus\{v\})} \mathbb{I}(r(v) = b) \right] \\ &= \mathbb{I}(r \in a_1) \left[(1-p_1)\mathbb{I}(x(v) = r(v)) + p_1 n_1(r, x^*)\mathbb{I}(r(v) = b) \right] \end{aligned}$$

下一步是仅当 $r \in a(r^*)$ 时才接受此状态；也就是说，以概率 $\mathbb{I}(r \in a(r^*))/\mathbb{I}(r \in a_1)$ 接受 r 作为 $a(r^*)$ 的一部分。如果接受，则新状态的权重为

$$w_2(r) = \mathbb{I}(r \in a(r^*)) \left[(1-p_1)\mathbb{I}(x(v) = r(v)) + p_1 n_1(r, x^*)\mathbb{I}(r(v) = b) \right].$$

如果发生拒绝，则必定是 $r(v) = b$ 的颜色，并且已经有一个邻居 v 具有该颜色。在这种情况下，执行与忽略移除相同的操作，并均匀随机地不放回搜索 v 的邻居，直到找到颜色为 b的邻居 w。禁止已搜索的邻居具有颜色 b，并将 w 固定为颜色 b。

回到接受的情况。令

$$p_1 = 1/d(x^*),$$

然后 $(1 - p_1) \leq p_1 n_1(r, x^*)$。

Let

$$p_2(r, x^*) = \mathbb{I}(r(v) \neq b) + \frac{d(x^*) - 1}{n_1(r, x^*)} \mathbb{I}(r(v) = b).$$

Then

$$w_2(r)p_2(r,x^*) = \mathbb{I}(r \in a(r^*)) \left[\left(1 - \frac{1}{d(x^*)}\right) \mathbb{I}(r(v) \neq b) + \frac{d(x^*)-1}{d(x^*)} \mathbb{I}(r(v) = b) \right]$$
$$= \mathbb{I}(r \in a(r^*)) \frac{d(x^*)-1}{d(x^*)}.$$

因此 $w_2(r)p_2(r,x^*) \propto w_{r^*}(r)$。所以，如果发生接受，权重就很好。

在拒绝的情况下，权重变为

$$w_2(r)(1-p_2(r,x^*)) = \mathbb{I}(r \in a(r^*)) \left[0 \cdot \mathbb{I}(r(v) \neq b) + \frac{n_1(r,x^*) - (d(x^*)-1)}{d(x^*)} \mathbb{I}(r(v) = b) \right]$$
$$= \mathbb{I}(r \in a(r^*)) \frac{n_1(r,x^*) - (d(x^*)-1)}{d(x^*)} \mathbb{I}(r(v) = b).$$

在这种拒绝情况下，状态必须有 $r(v) = b$，但它获得的权重与 $n_1(r,x^*) - (d(x^*) - 1)$ 成正比。$\mathbb{I}(r(v) = b)$ 这一部分很容易处理：将节点 v 冻结为颜色 b。

要理解剩下的表达式 $n_1(r,x^*) - (d(x^*) - 1)$，可以按指定顺序依次遍历 $U(x^*)$ 中的节点。每遇到一个邻居，要么它是一个此前 v 未见过、因而会被禁用的颜色，要么它不会禁用新的颜色。它之所以不会禁用新的颜色，可能是因为它与某个冻结颜色、禁用颜色或 $U(x^*)$ 中此前已检查过的邻居的颜色相同。

总的来说，表达式 $n_1(r,x^*) - (d(x^*) - 1)$ 统计的是遍历 $U(x^*)$ 时不会禁用新颜色的邻居数量。

理由如下。常数 $d(x^*)-1$ 表示：计入冻结邻居的颜色以及在当前情况下冻结邻居无法使用的禁用颜色后，可能可用的颜色数量，再减去 $U(x^*)$ 中邻居的实际数量。

这些邻居中的每一个 $U(x^*)$ 都可能禁用一种不同的颜色，因此在 x^* 下，节点 v 恰好还剩下 $d(x^*) - 1$ 种颜色可用。

但事实上，$n_1(x,\ x^*)$ 对于 x^* 有不同的颜色，其中 $n_1(x,\ x^*) \geq d(x^*) - 1$。因此，每个对 $n_1(x,\ x^*) - (d(x^*) - 1)$ 有贡献的邻居，都是没有阻挡新颜色，而是重复了已被阻挡颜色的邻居。

例如，假设 $k = 10, b = 10$，有一个邻居被冻结为颜色 3，而四个未冻结且未忽略的邻居分别是颜色 3、2、2、1。

这样，着色为 3 的邻居不会阻挡新的颜色，因为该颜色已经被冻结邻居阻挡。第一个着色为 2 的邻居会阻挡新的颜色，但第二个着色为 2 的邻居不会。最后，着色为 1 的邻居会阻挡新的颜色。因此，总共有两个邻居会阻挡新的颜色。

现在考察这个示例的表达式。这里 $n_1(x,\ x^*) = 10 - 3 - 1 = 6$ 由于颜色 4 到 9 可用于 $x(v)$。接下来，有 9 种颜色未被冻结邻居阻挡，并且有 4 个邻居既未冻结也未忽略，因此 $d(x^*) = 9 - 4 = 5$。因此$n_1(r,\ x^*) - (d(x^*) - 1) = 6 - (5 - 1) = 6 - 4 = 2$。

下一步是将该公式的责任份额分配给 $U(x^*)$ 中的每个邻居。着色为禁用颜色的邻居获得份额 1，着色为与冻结邻居相同颜色的邻居也获得份额 1。

假设三个邻居的颜色都是 4。那么它们对这个公式的贡献为 $3 - 1 = 2$。如果给这三个节点各分配 $2/3$ 的份额，那么它们的总贡献为 $(2/3) + (2/3) + (2/3) = 2$。

一般来说，在 $U(x^*)$ 中与其他邻居颜色相同、但颜色不是冻结颜色或禁用颜色的邻居，会获得 $(\ell - 1)/\ell$ 的份额，其中 ℓ 是 $U(x^*)$ 中颜色相同的邻居数。设 $h(w)$ 为每个节点的份额。

从邻居 w 中按与 $h(w)$成正比的概率选取一个。由于

$$\sum_{w \in U(x^*)} h(w) = n_1(r, x^*) - (d(x^*) - 1),$$

当节点 w 的颜色被冻结时，该状态的权重为

$$w_3(r) = \mathbb{I}(r \in a(r^*)) \frac{h(w)}{d(x^*)} \mathbb{I}(r(v) = b) \mathbb{I}(\text{neighbor } w \text{ frozen at } r(w)).$$

冻结这个节点会揭示它的颜色。如果它是禁用颜色或另一个冻结节点的颜色，那么 $h(w) = 1$，且权重是均匀的。

然而，如果这个节点是这些颜色之一，那么在其余邻居中至少有一个匹配项，也可能有更多。最佳做法是冻结 $U(x^*)$ 中所有共享此颜色的成员，然后将 v 重新冻结为此颜色，以表明 $U(x^*)$ 中没有其他成员具有此颜色，

最后，禁止 v 的邻居使用原来的颜色 b，因为这是该状态所处的子情况。

综上所述，移除禁用颜色的过程如下。

移除禁用（$x,\ x^*,\ v$）
输入状态 x，其索引为 x^*，其中 $x^*(v) = -b$

1. 如果 $(\exists t \in N(v))(x^*(t) = b)$，则令 $y(v) \leftarrow x(v)$，$y^*(V \setminus \{v\}) \leftarrow x^*(V \setminus \{v\})$，$y^*(v) \leftarrow \perp$，输出 (y, y^*) 并退出。3. 令 d 为在 x^*中未被 v 的冻结邻居使用的颜色数量（）减去未冻结或忽略的邻居数量，以概率 $1/d$，令 $r(v) \leftarrow b$；否则令 $r(v) \leftarrow x(v)$。4. 如果 $r(v) \neq b$，则输出 (r, r^*) 并退出。5. 如果 v 有一个颜色为 b的邻居，则在 v 的邻居中均匀地进行无放回搜索，直到找到这样的邻居 w。令 s^* 为与 r^* 相同的索引，但搜索过的、位于 w 之前的邻居均被禁止使用颜色 b，且 w 在 b 处冻结。输出 (r, s^*) 并退出。6. 令 n_1 为在x^*条件下使用 $x(V \setminus \{v\})$时节点 v 可用的颜色数量。以概率 $(d-1)/n_1$，输出 (r, r^*) 并退出。7. 对于 v的每个未冻结或未忽略邻居 w，设 $h(w)$为：若其与 v 的某个冻结邻居颜色相同，或其颜色为禁用颜色，则取 1；若 w 属于 v 的一个组，其中有 g 个未冻结或未忽略邻居且它们颜色均相同，则取 $(g-1)/g$。以与 $h(w)$成正比的概率选择一个节点 w。8. 如果 w 的颜色与某个冻结节点的颜色相同，或其颜色为禁用颜色，则令 q^*为与 r^* 相同的索引，但 v 和 w 分别冻结为其当前颜色；输出 (r, q^*) 并退出。9. 令 (t, t^*)为与 (r, r^*)相同的状态-索引对，但将 v 所有颜色与 w 相同的邻居冻结为其当前颜色，将 v 其余未冻结邻居的颜色设为禁用，并将 v 冻结为与 w相同的颜色。输出 (t, t^*) 并退出。

引理 2. 移除禁用 的权重函数由输出状态对应的输出索引给出。

证明。 证明（尽管更加复杂）沿用了与前面证明类似的思路。对于每个可能的结果索引，都必须检查最终得到的权重是否正确。

当输出出现在该步骤的第 1 行时， $x = y$ 和 $a(x^*) = a(y^*)$，因此 $w_{x^*}(x) = w_{y^*}(y)$，无需进一步考虑。

当输出在第 4 行或第 6 行包含 $y^* = r^*$ 时，回想一下， p_1 和 p_2之所以被选中，是为了确保 $w_{y^*}(y)$ 正确。

当输出在第 5 行包含 $y^* = s^*$ 时，搜索阻塞邻居的概率与前一个事实的证明中相同，这里的计算也完全相同。

当第 8 行的输出为 (r, q^*) 时，则 $h(w) = 1$，因为从 q^* 开始，新冻结的邻居 w 要么与 v 的另一个冻结邻居的颜色相同，要么与禁用颜色相同。

索引 q^* 将 v 冻结为颜色 b。因此，w_3 中的权重因子 $\mathbb{I}(r(v) = b)$ 成为关于 q^* 的函数 f_1，所以可以写为

$$w_{q^*}(r) = \mathbb{I}(r \in a(q^*)) \frac{1}{d(x^*)} f(q^*).$$

这使得分布在 $a(q^*)$ 上均匀。

在第 9 行的最后一种输出类型（t, t^*）中，由于所有与 w 颜色相同的节点在 t^* 中都被冻结，因此很容易计算 v 的具有该颜色的邻居数量。因此，对于函数 f_2，有 $h(w) = f_2(t^*)$。此外，由于 v 的所有其他未冻结邻居的颜色都是禁用颜色 b， $\mathbb{I}(r(v) = b)$ 是关于 t^* 的函数 f_3。这使得 w_3 等于

$$w_{t^*}(t) = \mathbb{I}(t \in a(t^*)) \frac{f_2(t^*)}{d(x^*)} f_3(t^*)$$

最终的分布在 $a(t^*)$ 上均匀。

□

3 定理 1 的证明

主算法中的每一步都写作：选择满足某种性质的 v，使得 $x^*(v)$.....但将索引写成

节点集合 V 的四分量划分，标记为 (V_{for}、V_{froz}、V_{ig}、V_{unr})，并附上一个颜色 b，该颜色是 V_{for} 中所有节点都禁用的颜色。这样一来，选择 v 就可以在常数时间内完成。

因此，每一步都会对 v、它的邻居，以及可能的某个邻居的邻居进行固定次数的修改。对于图的最大度数 Δ，这意味着每一步的执行时间至多为 $O(\Delta)$。

除了每一步进行的各种抛硬币操作外，有些步骤还会从 C中均匀选择一种颜色。此外，使用 $O(\Delta\log(\Delta))$比特，以均匀随机且不重复的方式搜索 v 中未被忽略且未冻结的邻居。因此，单步中的随机抽取只需 $O(\Delta\log(k))$ 个均匀随机比特即可完成。

因此，剩下的问题是：从所有节点都被忽略的状态开始，使一个节点变为不受限制，所需步数的期望值是多少?

如果当前索引中所有节点都被忽略，那么在**合法着色的RR算法**中，到达所有节点都不受限制的索引所需步数的期望值至多为 $O(\#V)$；当 $k > 3.637\Delta$时，每一步的执行时间为$O(\Delta)$。

令 T 表示算法终止所需的步数。

与这类证明中的常见做法一样，我们将使用一个定义在索引上的势函数 ϕ，并证明它在每一步都会减少 $\epsilon > 0$。也就是说，对于任意一个适当的节点 v，无论执行、x^*等于 y^* 时的**移除冻结**、**移除禁用**还是**移除忽略项**，都将证明

$$\mathbb{E}[\phi(\mathcal{A}(x, x^*))|\phi(x^*)] \leq \phi(x^*) - \epsilon.$$

鞅理论中的标准结果（加上每一步中 $\phi(x^*)$ 都有正概率减少固定量这一事实）便可得出

$$\mathbb{E}[T] \leq \phi(x_0^*)/\epsilon.$$

该函数为每个条件（禁用、冻结、忽略）赋予一个正权重。由于权重可以乘以任意正因子而不影响结果，因此可以将忽略节点的权重缩放为 1。

$$\begin{aligned}\phi(x^*) &= w_1\#(\{v : x^*(v) \in C\}) + w_2\#(\{v : x^*(v) \in -C\}) + \#(\{v : x^*(v) = 0\}) \\ &= w_1\#V_{\text{froz}} + w_2\#V_{\text{for}} + \#V_{\text{ig}}.\end{aligned}$$

因此，对于每个节点都被忽略的初始状态 x_0^*，$\phi(x_0^*) = \#V$。终止时，$\phi(x^*) = 0$。

现在设置权重。

移除冻结。 移除冻结使我们能够理解如何关联 w_1 和 w_2。请记住，移除一个冻结节点会产生一个忽略节点，并且可能额外产生最多 Δ个禁用节点。因此，令

$$w_1 = 1 + \Delta w_2 + \epsilon,$$

移除一个冻结节点所导致的预期变化至多为 $-\epsilon$。

现在考虑这个忽略节点。如果 $x(v)$与它的任何邻居都不匹配，就会移除一个忽略节点。但如果匹配，则平均至多会产生 $(\Delta - 1)/2$ 个禁用节点和一个冻结节点。

移除忽略项。每个忽略节点的颜色都是从 C中均匀选取的。因此，发生冲突的概率至多为 Δ/k，所以势函数的预期变化至多为

$$(-1)\frac{k-\Delta}{k} + \frac{\Delta}{k}\left[\frac{\Delta-1}{2}w_2 + w_1\right]$$

Let

$$\alpha = \frac{k-1}{\Delta},$$

那么它的上界为

$$-\frac{\alpha-1}{\alpha} + \frac{1}{\alpha}\left[\frac{3\Delta-1}{2}w_2 + \epsilon\right]$$

当.....时，这小于 $-\epsilon$

$$w_2 = \frac{2}{3\Delta-1}(\alpha - 1 - \epsilon(\alpha+1)) \tag{1}$$

对于 $0 < \epsilon < (\alpha-1)/(\alpha+1)$。

移除禁用

现在考虑在 b 移除禁用条件时会发生什么。在第 **1** 行中，该条件被直接移除，因此预期变化为 $-w_2$。

在第 **4** 行中，该条件也会被移除。此行以概率$1 - 1/d$ 激活，同时使禁用条件的数量减少 **1**。

以概率 $1/d$，尝试用禁用颜色 b重新着色 v 。

如果已有邻居被着色为 b，则会导致拒绝。搜索这个邻居平均会增加 $\Delta-1$ 个禁用节点，随后 w 被冻结，并且v 变为忽略。好的一面是， v 上的禁用条件被移除，因此势函数的平均变化为

$$\frac{1}{d}\left[-w_2+\frac{\Delta-1}{2}w_2+1+(1+\Delta w_2+\epsilon)\right]=\frac{1}{d}\left[\frac{3(\Delta-1)}{2}w_2+2+\epsilon\right].$$

如果没有邻居被着色为 b，那么拒绝的概率为 $1/n_1(x,\ x^*)\ \leq 1/(k-\Delta-1)$ $=(1/(\alpha-1))(1/\Delta)$，在这种情况下， v会被冻结（但至少不再被禁用。）

被冻结的未冻结且未忽略邻居的预期数量是多少？这类邻居至多有 Δ 个。某个特定邻居与其他 $\Delta-1$ 个邻居之一颜色相同的概率至多为 $(\Delta-1)/(k-\Delta-1)\ \leq 1/(\alpha-1)$。

只有当 v 的颜色被切换为 b 时，才会执行第 8 行和第 9 行；这种情况发生的概率为 $1/d$。在这种情况下，如果 v的邻居都没有颜色 b，平均会有多少个邻居被冻结?

考虑 v的某个特定邻居 w 。它被冻结的概率是多少？假设使用 x^* 选定采样着色的其余部分，但不包括 w 。其余邻居中，有 a_i 个颜色为 $i\in k$。那么， w 被冻结为颜色 i 的唯一可能情形是： $a_i>0$，发生拒绝， w 的颜色为i，并且选中了颜色 i 。注意， $a_i>0$ ，这可能是因为 v 有一个冻结邻居，也可能是因为有一个未冻结邻居。

w 具有颜色 i的概率至多为

$$\frac{1}{k-\Delta-1}$$

因为 w 至多有 Δ 个邻居，而且 w可能有一个禁用颜色。

令 h 为对邻居 w' （不包括 w）的 $h(w')$求和。如果 w具有颜色 i ，其中 $a_i>0$，则 $h(w')$之和会增加1。因此，在为 w 选定颜色 i 后，将颜色 i 冻结的概率为

$$\frac{h(w)}{h+1}.$$

拒绝的概率为 $(h+1)/n_1$，其中 n_1 是节点 v 在 x^*下的可用颜色数。因此，邻居 w v 的

在第 8 行或第 9 行被冻结的概率至多为

$$\frac{1}{k-\Delta-1}\cdot\frac{h(w)}{h+1}\cdot\frac{h+1}{n_1}=\frac{h(w)}{n_1(k-\Delta-1)}.$$

将这些关于所有邻居 w' 的预期变化上界求和，可得冻结邻居的期望数量至多为

$$\sum_{w'}\frac{h(w')}{n_1(k-\Delta-1)}=\frac{h+1}{n_1(k-\Delta-1)}\leq\frac{\Delta}{(k-\Delta-1)^2}=\frac{1}{(\alpha-1)^2\Delta}.$$

最后，如果发生拒绝， v 会被冻结，并可能产生多达 Δ 个禁用节点，因此冻结节点的数量增加 1。这使总变化（包括最初尝试重新着色 v 为 b 的概率 $1/d$ ）为

$$\frac{1}{d}\left[\frac{h+1}{n_1}\left[1+\Delta w_2\right]+\frac{1}{(\alpha-1)^2\Delta}\right]\leq\frac{1}{d}\left[\frac{1}{\alpha-1}\left[1+\Delta w_2\right]+\frac{1}{(\alpha-1)^2\Delta}\right].$$

这小于当存在一个着色为 b的邻居 v 时势函数的预期变化，因此该表达式占主导地位。

总体而言，此步骤中预期变化的上界 δ 为

$$\delta\leq\left(1-\frac{1}{d}\right)(-w_2)+\frac{1}{d}\left[\frac{3(\Delta-1)}{2}w_2+2+\epsilon\right].$$

要保证 $\delta\leq-\epsilon$ ，需要满足

$$-\epsilon\geq-w_2+\frac{1}{d}\left[\frac{3(\Delta+1)}{2}w_2+2+\epsilon\right].$$

当且仅当以下条件成立时，这才为真

$$d\geq\frac{(3/2)(\Delta-1)w_2+2+\epsilon}{w_2-\epsilon}.$$

Now

$$d\geq k-\Delta\geq\Delta(\alpha-1),$$

因此，当

$$\Delta(\alpha-1)=\frac{(3/2)(\Delta-1)w_2+2+\epsilon}{w_2-\epsilon}.$$

解这个方程可得

$$\epsilon = \frac{2\Delta\alpha^2 - 7\Delta\alpha - \Delta + 3\alpha - 1}{3\Delta^2\alpha - 3\Delta^2 + 2\Delta\alpha^2 - 4\Delta\alpha - \Delta + 3\alpha + 2}.$$

只要 $\epsilon > 0$，预期步数就有上界 n/ϵ。

在该表达式中，要使其随着 Δ 的增大而大于 0，必须满足 $2\alpha^2 - 7\alpha - 1 > 0$，或者 $\alpha > (1/4)(7 + \sqrt{57}) = 3.637\ldots$。

至此，定理 1 的证明完成。

参考文献

[1] S. Bhandari 和 S. Chakraborty。图中 k 着色的完美采样的改进界。在 第 *52* 届 *ACM SIGACT* 计算理论年度研讨会论文集中，第 631–642 页，2020年。[2] S. Chen、M. Delcourt、A. Moitra、G. Perarnau 和 L. Postle。通过线性规划随机采样着色的改进界。在第 *30* 届 *ACM-SIAM* 离散算法年度研讨会论文集中，第 2216–2234 页，2019年。[3] J. A. Fill 和 M. Huber。随机性回收器：一种完美采样的新技术。在 第 *41* 届 计算机科学基础年度研讨会论文集中，第 503–511 页，2000年。[4] F. Guthrie。关于地图着色的注记。爱丁堡皇家学会会刊，10:727–728，1880年。[5] M. L. Huber。完美模拟。CRC 出版社，2016年。[6] M. Huber。精确采样和近似计数技术。在第 *30* 届 *ACM* 理论计算年度研讨会论文集中，第 31–40 页，1998年。

`https://doi.org/10.1145/276698.276709`。[7] V. Jain、A. Sah 和 M. Sawhney。在图中完美采样 $k \geq (8/3+o(1))\Delta$ 着色。在 第 *53* 届 *ACM SIGACT* 计算理论年度研讨会论文集中，第 1589–1600 页，2021年。[8] M. Jerrum。计算多变量多项式值的复杂性。信息与计算，72(2):155–162，1987年。

[9]M. Jerrum。估计低度图中 k 着色数量的一个非常简单的算法。*Random Structures & Algorithms*，7(2):157–165，1995年。[10] M. Jerrum、L. G. Valiant 和 V. V. Vazirani。从均匀分布中随机生成组合结构。理论计算机科学，43:169–188，1986年。https://doi.org/10.1016/0304-3975(86)90174-X。[11] R. M. Karp。组合问题之间的可归约性。载于 R. E. Miller、J. W. Thatcher 和 J. D. Bohlinger 主编的 *Complexity ofComputer Computations*，第 85–103 页。Springer US，美国马萨诸塞州波士顿，1972年。https://doi.org/10.1007/978-1-4684-2001-2_9。[12] J. von Neumann。与随机数字相关的各种技术。*Applied Mathematics Series*，12:36–38，1951年。[13] J. G. Propp 和 D. B. Wilson。使用耦合马尔可夫链进行精确采样及其在统计力学中的应用。*Random Structures & Algorithms*，9(1–2):223–252，1996年。[14] J. G. Propp 和 D. B. Wilson。如何从一般马尔可夫链中获得完全随机的样本并生成有向图的随机生成树。*Journal of Algorithms*，27(2):170–217，1998年。[15] E. Vigoda。着色采样的改进界。在 第*40*届计算机科学基础年会中，第 51–59 页，1999年。